https://folk.ntnu.no/michaeng/tdt4205_21/
michael.engel@ntnu.no

Theoretical exercises
Spring 2021

# Theoretical Exercises 5

**Assembler**

**Please submit solutions on Blackboard by Friday, 26.03.2021 14:00h**

*Hint:* You can use the x86-64 cheat sheet (https://www.cs.tufts.edu/comp/40/docs/x64_cheatsheet.pdf) to look up assembler instruction details.

## 5.1 x86-64 code analysis (4 points)

Consider the following x86-64 assembler code function, compiled from C:

```
Disassembly of section .text:

0000000000000000 <foo>:
   0: 89 f0                mov    %esi,%eax
   2: 85 f6                test   %esi,%esi
   4: 7e 0f                jle    15 <foo+0x15>
   6: 85 d2                test   %edx,%edx
   8: 74 0c                je     16 <foo+0x16>
   a: 31 d2                xor    %edx,%edx
   c: 01 f8                add    %edi,%eax
   e: 83 c2 01             add    $0x1,%edx
  11: 39 d0                cmp    %edx,%eax
  13: 7f f7                jg     c <foo+0xc>
  15: c3                   retq
  16: 29 f8                sub    %edi,%eax
  18: 83 c2 01             add    $0x1,%edx
  1b: 39 d0                cmp    %edx,%eax
  1d: 7f f7                jg     16 <foo+0x16>
  1f: c3                   retq
```

a. How many parameters does the function take? Which instructions indicate this (give the instruction address)?

b. Does the code of the function include an if statement? How did you find this out?

c. Does the code of the function include a loop? How did you find this out?

d. Does the function return a value?

## 5.2 Decompile! (2 points)

The following x86-64 assembly code is given:

```
f:
        movl a, %eax
        movl b, %edx
        andl $255, %edx
        subl %edx, %eax
        movl %eax, a
        retq
```

a. Give equivalent valid C code that would compile *without warnings* to this assembler code function. Assume the declaration `extern unsigned a, b;`. Don't run a C compiler to obtain the result.

b. Find two *different* versions of C code that compile to the above code. One of these should have a different function signature than the ones you described already.

## 5.3 Data types (5 points)

For each of the following x86-64 assembler instructions, give the type of the data object that is most likely to be accessed by this code. Indicate the reason for your answer.

- `movzbl %al, %eax`
- `movl -28(%rbp), %edx`
- `movsbl -32(%rbp), %eax`
- `movl (%rdx,%rax,4), %eax`
- `movzbl 4(%rax), %eax ; movsbl %al, %eax`